Macros for Financial Applications
Prepared by Pamela Peterson Drake, James Madison University
Last revision: November 26, 2010

Table of contents

Contents
The basics ........................................................................................................................................... 1
  What is a macro? ............................................................................................................................... 1
  Recording a macro v. using VBA ................................................................................................. 1
  Include “Developer” in the ribbon ............................................................................................... 2
  Save as a Microsoft Excel macro-enabled workbook ................................................................. 2
  Enable macros ............................................................................................................................... 2
How to record a macro .................................................................................................................... 3
How to use VBA to create a macro .................................................................................................. 6
  VBA code highlights ...................................................................................................................... 8
VBA Examples ................................................................................................................................... 9
  A simple macro to sum two numbers ............................................................................................ 9
  Convert to a currency format ......................................................................................................... 10
  Using loops .................................................................................................................................... 11
  Run a macro with a button ........................................................................................................... 12
  Run a macro using a shortcut key .............................................................................................. 13
  Have a macro talk back to you .................................................................................................... 14
  Replace #N/A with blanks ........................................................................................................... 14
  Calculate present values ............................................................................................................. 15
  Calculate the present and future values .................................................................................... 16
Help hints (and warnings) ............................................................................................................. 17
Commands in a macro .................................................................................................................... 18
  Formatting ..................................................................................................................................... 18
  Other commands .......................................................................................................................... 18
Index ................................................................................................................................................ 19
The basics

What is a macro?
A macro is a snippet of code that instructs the software (e.g., Microsoft Excel) to perform an operation or a series of operations. Macros are most useful when you have repetitive tasks or when you want to reuse a set of operations for different worksheets or workbooks.

Recording a macro v. using VBA
You can create a Macro by recording operations that you code directly in a worksheet, or by using a set of Visual Basic code and programming these operations. In either case, you need to enable macros in Microsoft Excel.

**RECORDING A MACRO**
1. Save the workbook as a Macro-enabled workbook
2. Alter Macro security setting to allow Macros
3. Start recording the Macro
4. Perform calculations in the worksheet that you want to record within the Macro
5. Stop recording
6. Use Macro, applying the CTRL+ key or by using Macros, select the Macro, and then Run

**USING VBA**
1. Save the workbook as a Macro-enabled workbook
2. Alter the Macro security setting to allow Macros
3. Go to the Visual Basic program via Developer and then Insert > Module
4. Enter your VBA code
5. Run your VBA code to make sure it works
6. Save the Macro (File > Save)
Include “Developer” in the ribbon

If Developer does not appear on the ribbon in Excel, go to Excel Options and in the Popular section, check the box for “Show Developer tab in the Ribbon”:

Save as a Microsoft Excel macro-enabled workbook

When you begin your project, be sure to save the workbook as Macro-Enabled. This is required.

Enable macros

Using the Developer portion of the Ribbon,
check the appropriate setting to enable all macros:

How to record a macro

Recording a macro is quite simple: you start recording, perform the operation(s), and then you stop recording. To record a Macro, simply click on the Record Macro in the Developer Ribbon:

When you record a macro, you will be asked to

- name the macro,
- provide the short-cut key,
- identify where the macro should be stored, and
- provide a description of the macro (optional, but useful)
Clicking on OK and returns you to the worksheet. Place the cursor in the cell (or highlight a group of cells) that you want the calculation and then type in the short-cut keys (e.g., CTRL v):

If you want to run the macro again, click on the link to macros in the ribbon, select the macro in the list of macro names, and then Run:
This produces:
How to use VBA to create a macro

You can use VBA to create a macro, which allows you to produce and test a set of code to perform functions. An advantage of creating such a macro is that you can store the Macro and use it on many different worksheets.

The first step is to set up the destination worksheet, e.g.,:

Clicking on the link to Visual Basic in the Developer Ribbon,

you enter Visual Basic:

When you Insert > Module, a blank workspace is created:
You then place your code in the workplace:

You run the program by either using the green arrow or Run in the ribbon, which produces:

If your program does not run, you can “Step in” to your program using F8, seeing each step and diagnosing any programming errors.
Hint: An easy way to learn about coding using VBA is to record a Macro and then Edit that Macro in Visual Basic, which will produce the code behind your recorded operations.

VBA code highlights

1. The VBA coding will start with Sub and end with end Sub. You name the Macro using the Sub command. For example, if you name your macro “Tree”,

   ```vba
   Sub Tree()
   End Sub
   ```

   The macro editor will add the ( ) after the name and will automatically produce the End Sub at the end of the coding.

2. You can refer to cells in a worksheet using a number of methods. The easiest is probably using Cells(row,column).

3. You can execute repetitive tasks using the For or the Do approaches. If, for example, you want to add values from columns B and C, placing the results in column A, for 10 observations in rows 1 through 10, you have a couple ways to do this:

   **Using For**
   ```vba
   For J = 1 to 10
   Cells(J,1)=Cells(J,2)+Cells(J,3)
   Next
   ```

   **Using Do While**
   ```vba
   J = 1
   Do While J < 10
   Cells(J,1)=Cells(J,2)+Cells(J,3)
   J=J+1
   Loop
   ```
VBA Examples

A simple macro to sum two numbers

This macro adds two values and places the sum, in bold, in a third cell.

```vba
Sub sum_two()
    ' sum_two Macro
    ' Keyboard Shortcut: Ctrl+s
    ActiveCell.FormulaR1C1 = "=SUM(R[-2]C:R[-1]C)"
    Range("A3").Select
    Selection.Font.Bold = True
End Sub
```

- The macro name is **sum_two**, which is, essentially, a subroutine that adds two numbers and then changes the value to a bold font.
- **End Sub** indicates the end of the subroutine.
- The lines with the single quotation are comment lines, which are not processed. Removing these would not change the macro.
- **ActiveCell.FormulaR1C1** refers to the cell that the cursor is in when the macro is run in terms of row and column notation.
- **"=SUM(R[-2]C:R[-1]C)"** instructs Excel to sum the values from two rows up to one row up (using negative values to indicate previous rows).
- **Range("A3").Select** indicates the cell to select, and then **Selection.Font.Bold = True** indicates that the selected cell should have a bold font.

Once you have created the macro for your worksheet (or workbook), you can use the keyboard shortcut to trigger the macro when the cursor is in any cell.
Convert to a currency format

This macro converts a value into a currency formatted value.

```
Sub dollar_sign()
    ' dollar_sign Macro
    ' Convert to currency
    ' Keyboard Shortcut: Ctrl+d
    'ActiveCell.NumberFormat = "$#,##0.00"
End Sub
```

- The macro name is `dollar_sign`
- The comment `Convert to currency` is the result of adding a description of a macro.
- `ActiveCell.NumberFormat = "$#,##0.00"` indicates the currency format for the cell in which the cursor lies.
- Adding the line `ActiveCell.Font.Bold = True` below the other ActiveCell command will result in the cell value having a bold font.
Using loops

This macro performs a function that is repeated until a stop condition is satisfied.

Sub Concat()
Sheets("Sheet2").Select
    Range("C1").Select

    Do Until Selection.Offset(0, -2).Value = ""
        Selection.Value = Selection.Offset(0, -2).Value & ", " & Selection.Offset(0, -1)
        Selection.Offset(1, 0).Select
    Loop
    Range("A1").Select
End Sub

- The macro name is Concat.
- This is a Do loop, which basically means that the code between the Do and the Loop is repeated until the stop condition is satisfied.
- The stop condition for this Do loop is that when the cell in the column two columns to the left (hence, the -2) is blank (" ").
- This Do loop happens to concatenate the data two columns to the left in the same row (0,-2), with the data in one column over in the same row (0,-1), with a comma and a space (", ") in between.
- The cursor returns to the cell indicated with the Range("A1").Select.

Before the macro

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Last</td>
<td>First</td>
</tr>
<tr>
<td>2</td>
<td>Miller</td>
<td>Ed</td>
</tr>
<tr>
<td>3</td>
<td>Modigliani</td>
<td>Franco</td>
</tr>
<tr>
<td>4</td>
<td>Sharpe</td>
<td>William</td>
</tr>
<tr>
<td>5</td>
<td>Markowitz</td>
<td>Harry</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

After the macro

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Miller</td>
<td>Ed</td>
<td>Last, First</td>
</tr>
<tr>
<td>2</td>
<td>Modigliani</td>
<td>Franco</td>
<td>Miller, Ed</td>
</tr>
<tr>
<td>3</td>
<td>Modigliani, Franco</td>
<td>Sharpe, William</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Sharpe</td>
<td>William</td>
<td>Sharpe, William</td>
</tr>
<tr>
<td>5</td>
<td>Markowitz</td>
<td>Harry</td>
<td>Markowitz, Harry</td>
</tr>
</tbody>
</table>
Run a macro with a button

The first step is to create your macro. Once you have created the macro, you can add a button to use to run the macro.

Creating the button:
1. Using the Developer ribbon, Insert a Form control button.

2. When the dialogue box pops up to Assign Macro, select the macro you want to associate with the button.

3. Use the format control (right click on the button) so that the button fits your style.
4. Click on the button to run the macro.
Run a macro using a shortcut key

When you record a macro, you identify the shortcut key before you record the macro. However, when you create your own macro, you need to assign the shortcut key by selecting “Options” and then specifying the key.
Have a macro talk back to you

You can create a message box using the message box, `MsgBox`, command.

Sub talkback()
    Sheets("Sheet3").Select
    Range("A1").Value = 1234
    MsgBox "The result is " & Range("A1").Value
End Sub

- The macro’s name is `talkback`.
- The macro involves Sheet 3
- The value place in cell A1 is 695.
- The `MsgBox` command reports the value in cell A1.

Replace #N/A with blanks¹

This code will replace values with values, and replace #N/A with blanks. Note: because of the replacement, formulas will be replaced by values.

Sub CommandButton1_Click()
    Dim count As Integer
    'paste values
    Range("A1:BZ150").Copy
    'replace ">#N/A" with 
    SearchOrder:=xlByRows, MatchCase:=False, SearchFormat:=False, _
    ReplaceFormat:=False
    'unique number
    count = 1
    For i = 1 To 150
        If Trim(Range("b" & i).Value) = "" Then
            Range("b" & i).Value = count
            count = count + 1
        End If
    Next i
End Sub

¹ Modified from: VBWire | VB forums
Calculate present values

for a range of interest rates, given a lump-sum, number of periods, and the periodic cash flow

This Macro is useful for calculating the present value for specified interest rates, given the future value, the number of payments, and the periodic cash flow.

Some notes about this Macro:

- The present value function refers to the following arguments:
  PV(interest rate, number of periods, periodic cash flow, future value).
  - The interest rate is one in the list in cells A9 through A29;
- The number of periods is in cell B5,
- the periodic cash flow is in B6,
- the future value is in B4.

“Cells” refers to the row and column location of a cell. For example, Cells(3,4) is cell D3, and Cells (4,3) is cell C4.

```
Sub PVMacro()
    For x = 9 To 29
        Cells(x, 2) = PV(Cells(x, 1), Cells(5, 2), Cells(6, 2), Cells(4, 2)) * -1
        Cells(x, 2) = Format(Cells(x, 2), "Currency")
    Next
End Sub
```
Calculate the present and future values
for a range of interest rates, given a lump-sum, number of periods, and the periodic cash flow

This Macro is useful for calculating present and future values over a range of interest rates. Some notes on this Macro:

- The range of interest rates is in the worksheet in cell B9 (the minimum) and B10 (the maximum)
- The lump-sum is in B4.
- The amount of the periodic cash flow is in B5
- The number of periods is in B6
- The output is in columns D, E and F, with the number of rows dependent on the requested interest rate range. The first row of results is the third row.

Sub TVM()

X = (Cells(9, 2) * 100)
Y = (Cells(10, 2) * 100)
Z = Y - X + 1
i = Cells(9, 2)

For m = 3 To (Z + 2)
    Cells(m, 4) = i
    Cells(m, 5) = PV(Cells(m, 4), Cells(6, 2), Cells(5, 2), Cells(4, 2)) * -1
    Cells(m, 6) = FV(Cells(m, 4), Cells(6, 2), Cells(5, 2), Cells(4, 2)) * -1
    i = i + 0.01
Next

End Sub
Help hints (and warnings)

1. Test your macro on a spreadsheet that doesn’t matter.
2. Remember, that if you assign a short-cut key, this shortcut key will replace Excel’s normal shortcut key for the worksheet or workbook -- which can produce some unpleasant surprises.
3. Build a workbook with all your handy macros and open this workbook when you want to use the macro.
4. Place your macros in a folder and make that folder a “trusted” folder in Microsoft’s security.
5. Design macros to be flexible with respect to cells. Refer to active cells, which are ones that you highlight.
6. Macros run on the active workbook, so be sure to have the correct workbook active when you use your macro.
7. Don’t reinvent the wheel: look for macro coding that is available from colleagues, the web, etc. Just be sure that you have the right to the coding and that you test it -- don’t just trust others to get it right.
8. When in doubt or when you can’t remember the coding, record a macro with a similar function and then check out the code.
# Commands in a macro

## Formatting

<table>
<thead>
<tr>
<th>Command</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ActiveCell.Font.Bold = True</code></td>
<td>Bold font</td>
</tr>
<tr>
<td><code>ActiveCell.NumberFormat = &quot;$#,##0.00&quot;</code></td>
<td>Currency format</td>
</tr>
<tr>
<td><code>ActiveCell.Font.Italic = True</code></td>
<td>Italic font</td>
</tr>
</tbody>
</table>

## Other commands

<table>
<thead>
<tr>
<th>Command</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>Sheets(&quot;Data&quot;).Select</code></td>
<td>Access the worksheet named Data</td>
</tr>
<tr>
<td><code>MsgBox &quot;Hi there&quot;</code></td>
<td>Generate a pop-up box</td>
</tr>
<tr>
<td><code>Sheets(&quot;Sheet1&quot;).Select</code></td>
<td>Identify which sheets to apply the macro to</td>
</tr>
<tr>
<td><code>ActiveCell.Offset(1,0).Select</code></td>
<td>Move one cell down (one more row) from the selected cell</td>
</tr>
<tr>
<td><code>Range(&quot;C3&quot;).Offset(0,-1).Select</code></td>
<td>Move one column left from the column of cell C3</td>
</tr>
<tr>
<td><code>Range(&quot;C3&quot;).Offset(1,0).Select</code></td>
<td>Move one row down from the row of cell C3</td>
</tr>
<tr>
<td><code>Range(&quot;A1:E10&quot;).Select</code></td>
<td>Select a range of cells</td>
</tr>
<tr>
<td><code>Range(&quot;A1&quot;).Select</code></td>
<td>Select a single cell</td>
</tr>
<tr>
<td><code>Range(&quot;A1,A5,B1:B10&quot;).Select</code></td>
<td>Select non-contiguous cells</td>
</tr>
<tr>
<td><code>Selection.AutoFilter</code></td>
<td>Shows or hides auto filters</td>
</tr>
</tbody>
</table>
Index

ActiveCell, 9, 10
ActiveCell.Font.Bold = True, 10, 18
ActiveCell.Font.Italic = True, 18
ActiveCell.NumberFormat, 10
ActiveCell.NumberFormat = ",##0.00", 18
Assign Macro, 12
Button, 12
Developer, 2
Developer ribbon, 6
End Sub, 9
Form control button, 12
Macro, 1

macro-enabled, 2
Message box, 14
MsgBox, 14, 18
Range("A1").Select, 11
Range("C3").Offset(0,-1).Select, 18
Record a macro, 3
Selection.AutoFilter, 18
Selection.Font, 9
Sheets("Data").Select, 18
Short-cut key, 3, 13
Sub, 8
VBA, 6